Towards Automated Computer Vision: Analysis of the AutoCV Challenges 2019

Zhengying Liu, Zhen Xu, Sergio Escalera, Isabelle Guyon, Julio C S Jacques Junior, Meysam Madadi, Adrien Pavao, Sebastien Treguer, Wei-Wei Tu

Towards Automated Computer Vision: Analysis of the AutoCV Challenges 2019

Zhengying Liu[a,1], Zhen Xu[b,c,1,*], Sergio Escalera[d,e,f], Isabelle Guyon[a,f], Julio C. S. Jacques Junior[g,d], Meysam Madadi[d], Adrien Pavao[a], Sebastien Treguer[h,f], Wei-Wei Tu[c,f]

[a] *UPSud/INRIA, Université Paris-Saclay, 91400 Orsay, France*
[b] *Ecole Polytechnique, 91120 Palaiseau, France*
[c] *4Paradigm, Beijing, China*
[d] *Computer Vision Center, Spain*
[e] *Universitat de Barcelona, Spain*
[f] *ChaLearn, USA*
[g] *Universitat Oberta de Catalunya, Spain*
[h] *La Paillasse, France*

ABSTRACT

We present the results of recent challenges in Automated Computer Vision (AutoCV, renamed here for clarity AutoCV1 and AutoCV2, 2019), which are part of a series of challenge on Automated Deep Learning (AutoDL). These two competitions aim at searching for fully automated solutions for classification tasks in computer vision, with an emphasis on any-time performance. The first competition was limited to image classification while the second one included both images and videos. Our design imposed to the participants to submit their code on a challenge platform for blind testing on five datasets, both for training and testing, without any human intervention whatsoever. Winning solutions adopted deep learning techniques based on already published architectures, such as AutoAugment, MobileNet and ResNet, to reach state-of-the-art performance in the time budget of the challenge (only 20 minutes of GPU time). The novel contributions include strategies to deliver good preliminary results at any time during the learning process, such that a method can be stopped early and still deliver good performance. This feature is key for the adoption of such techniques by data analysts desiring to obtain rapidly preliminary results on large datasets and to speed up the development process. The soundness of our design was verified is several respect: (1) Little overfitting of the on-line leaderboard providing feedback on 5 development datasets was observed, compared to the final blind testing on the 5 (separate) final test datasets, suggesting that winning solutions might generalize to other computer vision classification tasks; (2) Error bars on the winners' performance allow us to say with confident that they performed significantly better than the baseline solutions we provided; (3) The ranking of participants according to the **any-time metric** we designed, namely the Area under the Learning Curve, was different from that of the **fixed-time metric**, *i.e.* AUC at the end of the fixed time budget. We released all winning solutions under open-source licenses. At the end of the AutoDL challenge series, all data of the challenge will be made publicly available, thus providing a collection of uniformly formatted datasets, which can serve to conduct further research, particularly on meta-learning.

Keywords: Computer Vision, AutoML, Deep Learning

(this paper is under review at Pattern Recognition Letters)

*Corresponding author
e-mail: zhen.xu@polytechnique.edu (Zhen Xu)
[1] Equal contribution. The other authors are ordered alphabetically.

While Machine Learning (ML) keeps delivering impressive novel applications in our daily lives, it is still facing enormous challenges, preventing its more universal deployment by users having direct needs but no time or resources to hire ML experts. In fact, even for ML experts, effectively tuning hyper-parameters is still a daunting task, particularly for Deep Learning models, let alone addressing higher level aspects of model design, including problem definition, experimental design, data collection, preprocessing, design of metrics, computation of error bars, detection of bias in data, etc. Certainly, automating the entire modeling pipeline is still a far reaching goal, but the challenges we present in this paper allowed us to make great strides. We present here the results of the two first editions of the Automated Deep Learning (AutoDL) challenge series, addressing Automated Computer Vision (AutoCV). With the solutions provided (and open-sourced) by the winners, users must only preprocess data to horseshoe-fit them in a generic tensor format to have automated algorithms train and test Deep Learning neural networks for them. The problems addressed are image or video classification, in an amazingly broad range of application domains (medical imaging, object or gesture classification, satellite imaging, to name a few). Besides saving human effort, the benefit of such automated solutions include reproducibility and accountability, freeing us potentially from the variability of human solutions and possibly increasing reliability.

The AutoDL challenge series in part of a larger effort on Automated Machine learning (AutoML) with **code submission** in which the solutions of participants are blind tested (on the Codalab challenge platform). In Liu et al. (2019), Liu and Xu introduce a mathematical formulation of AutoML, defining a hierarchy ($\alpha-\beta-\gamma$ levels) of *supervised learning problems* lending themselves to different types of challenge settings. The levels differ in the amount of information made available to challenge participants and to their "autonomous agent" (code submitted). Classical "result submission" challenges *à la* Kaggle are not covered in this hierarchy, but as in those challenges, the final test set labels are never revealed to the participants: they are used by scoring programs residing on the challenge platform to evaluate entries. Briefly, the levels consist of: **α-level** imposing that submitted code include **trained models with solely a test method**, hence training data and unlabeled test data are provided to the participants; **β-level** imposing that submitted code include **training and test methods**, hence both training and unlabeled test data are solely available to the autonomous agent, NOT to the participants; **γ-level** imposing that submitted code include also a **meta-learn method** to be used for training the autonomous agent to deliver a **β-level** algorithm. Thus while in α and β-level settings the agent is trained and tested independently on all tasks, in the γ-level setting, the autonomous agent can learn from past tasks to perform better on next tasks (**meta-learning**). One of the insights drawn from the Liu-Xu framework is that hyperparameter optimization (HPO) methods (*e.g.* Neural Architecture Search Zoph and Le (2016); Pham et al. (2018)) do not really address fully the AutoML problem more than "classical" machine learning algorithms, in the sense that they remain at the β-level (no meta-learning). In the AutoCV challenge setting, we encourage meta-learning by providing sample datasets and having a challenge in two phases, thus testing participants' solutions of a variety of tasks. However, we do not impose to participants to deliver a "meta-learn" method and final testing is performed on the five tasks of the challenge independently (of one another). In that sense, our challenge is still a β-level challenge. In the past, our AutoML3 challenge (Life-Long-Machine-Learning with drift) was at the γ-level.

In all of our AutoML challenges we seek to enforce learning within a fixed time budget (in this case, 20 minutes per dataset on a single GPU) and fixed computer resources (in this case, the compressed code had to be under 300 MB, which allowed participants to submit pre-trained models, and we ran submission on Google Cloud NVIDIA Tesla P100 GPUs, running CUDA 10 with drivers cuDNN 7.5 coupled with 4 vCPUs, with 26 GB of memory, 100 GB disk). One particularity of AutoDL challenges compared to previous AutoML challenges is that we seek to enforce **any-time learning**. By this we mean that our metric encouraged participants to deliver predictions that are regularly saved and are reasonably good early on in the learning process. Specifically, the metric of evaluation is the Area under the Learning Curve (ALC), see Section 2.

We summarize basic facts in Table 1. While most of our challenges are run in two phases (a feedback phase with immediate feedback on a leaderboard on N practice datasets and a final test phase with a single evaluation on N final test datasets), in AutoCV1, we evaluated the participants on the results of the feedback phase, to make it slightly easier. However, we ran privately a final test phase of which we report here the results. Since the 5 AutoCV1 final phase datasets were not disclosed, we re-used some in subsequent phases. AutoCV2 was run regularly in 2 phases. Even practice datasets during the feedback phase were not revealed to the participants (they were solely visible to their "autonomous agent"). To allow them to develop their code, we provided them with sample "public" datasets. Additionally a starting kit in Python with TensorFlow and PyTorch interfaces was provided, including sample code submissions.

1. Data

In the AutoCV challenges, **raw data** are provided to participants (images or videos) formatted in a uniform tensor manner (namely TFRecords, a standard generic data format used by TensorFlow[2]). For images with native compression formats (e.g. JPEG, BMP, GIF), we directly use the bytes. Our data reader decodes them on-the-fly to obtain a 3D tensor. Video files in mp4/avi format (without the audio track) are used in a similar manner. For practical reasons, datasets were kept under 2GB, which required sometimes reducing image resolution, cropping, and/or downsampling videos. We made sure to include application domains in which the scales varied a lot (from microscopic level to satellite images, going through human-scale level).

We formatted 25 datasets (15 image and 10 video datasets, see Table 2). Note that 3 datasets (Loukoum, Apollon, Ideal) are used twice in different occasions. All tasks are supervised

[2]To avoid privileging a particular type of Deep Learning platform, we also provided a data reader to convert to PyTorch format.

Table 1: ***Basic facts on AutoCV 1 & 2 challenges****.*

Challenge	Collocated with (2019)	Begin date 2019	End date 2019	#Teams	#Submissions	#Phases	#Datasets		
							public	feedback	final
AutoCV1	IJCNN	May 1	Jun 29	102	938	1	5	5	5
AutoCV2	ECML PKDD	July 2	Aug 20	34	336	2	3	5	5

classification problems, *i.e.* data samples are provided in pairs $\{X, Y\}$, X being an input 4D tensor of shape (time, row, col, chnl) and Y a target binary vector (withheld from in test data). Since we intend to re-use those datasets in an upcoming challenge on life-long meta-learning, we do not provide further details.

2. Evaluation Metrics

AutoCV challenges encourage **any-time learning** by scoring participants with the Area under the Learning Curve (ALC) (Figure 1). The participants can train in increments of a chosen duration (not necessarily fixed) to progressively improve *performance*, until the time limit is attained. Performance is measured by the NAUC or *Normalized* Area Under ROC Curve (AUC) $NAUC = 2 \times AUC - 1$ averaged over all classes. Multi-class classification metrics are not being considered, *i.e.* each class is scored independently. Since several predictions can be made during the learning process, this allows us to plot learning curves, *i.e.* "performance" as a function of time. Then for each dataset, we compute the Area under Learning Curve (ALC). The time axis is log scaled to put more emphasis on the beginning of the curve since we wanted to force participants to develop techniques that can climb performance fast because when they will be exposed to "big data" this will matter a lot more. Finally, an overall rank for the participants is obtained by averaging their ALC ranks obtained on each individual dataset, in a given phase.

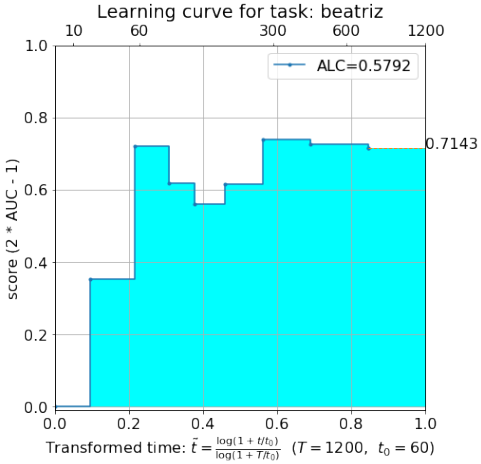


Fig. 1: **Example of learning curve:** *performance as a function of time. The x-axis corresponds to timestamp but normalized to [0,1]. The Area under the Learning Curve (ALC) is the challenge metric. We integrate with the rectangle method (i.e. with step function) because the trapeze method would give an advantage to "lazy" participants who do not save results regularly. This figure shows an example of possible over-fitting in which the participant could have stopped further training earlier.*

3. Baselines

We introduced baseline methods with varied complexity and computer resource requirements. For debug purposes, we provided a trivial baseline (Baseline 0) returning all-zero predictions. Baseline 0 always gets 0 NAUC score (hence 0 ALC as well), hence we do not include it in the analyses. Baseline 1 is a linear model and Baseline 2 a (self-scaling) convolutional neural network (CNN). We also (privately) used a pre-trained Inception network as baseline. Only Baseline 0, 1, and 2 were provided to participants.

3.1. Baseline 1: Linear Classifier with Basic Scheduling

This method uses a single layer neural network. In the feed-forward phase, the input tensor is flattened then fully connected to the output layer, with a sigmoid activation. During training, it uses a cross entropy loss (as in logistic regression) and the Adam optimizer Kingma and Ba (2014), with default learning rate. The batch size is fixed to 30 for both training and test. If the input shape is variable, some preprocessing procedure is required: we simply resize all images to a fixed shape 112×112 (the number of channels is always fixed). When the number of frames (time axis) is variable, we simply sample 10 consecutive frames at random, both for training and testing.

As we are in an any-time learning setting in AutoCV challenges, we need a scheduling strategy to make good predictions quickly. For this baseline, we use following scheduling strategy: At the beginning, the algorithm trains the neural network for $s = 10$ steps. An estimation of time used per step is computed. Then the number of training steps doubles ($s \leftarrow 2s$) at each train/test call and the algorithm computes the duration required for this number of training steps using the estimation. This estimated duration is then compared to remaining time budget (sent by ingestion program). If there is still enough time, the program calls training again; otherwise, it pro-actively stops.

3.2. Baseline 2: Self-scaling 3D CNN

Baseline 2 is the strongest public baseline that we provided to participants (and participants needed to beat this baseline in order to enter final phase of AutoCV2). Compared to baseline 1, the only difference is the neural network architecture, which is determined according to the tensor shape of the input examples. More concretely, we repeatedly apply a 3D convolutional layer followed by a 3D max-pooling layer, until the number of neurons of the hidden layer is less than a pre-defined number (e.g. 1000), then we apply a fully connected layer for classification (after flattening the hidden layer). The filter shape of 3D CNN layers

Table 2: **Datasets used in AutoCV 1 & 2 challenges.** *'hwr' means handwriting recognition, 'img' image, 'vid' video, and 'var' variable size.*

#	Dataset	Auto CV#	Phase	Domain	Type	Class num.	Sample number		Tensor dimension			
							train	test	time	row	col	chnl
1	Munster	1	public	hwr	img	10	60000	10000	1	28	28	1
2	Chucky	1	public	objects	img	100	48061	11939	1	32	32	3
3	Pedro	1	public	people	img	26	80095	19905	1	var	var	3
4	Decal	1	public	aerial	img	11	634	166	1	var	var	3
5	Hammer	1	public	medical	img	7	8050	1965	1	600	450	3
6	Ukulele	1	feedback	hwr	img	3	6979	1719	1	var	var	3
7	Caucase	1	feedback	objects	img	257	24518	6089	1	var	var	3
8	Beatriz	1	feedback	people	img	15	4406	1094	1	350	350	3
9	Saturn	1	feedback	aerial	img	3	324000	81000	1	28	28	4
10	Hippocrate	1	feedback	medical	img	2	175917	44108	1	96	96	3
11	Loukoum	1	final	hwr	img	3	27938	6939	1	var	var	3
12	Tim	1	final	objects	img	200	80000	20000	1	32	32	3
13	Apollon	1	final	people	img	100	6077	1514	1	var	var	3
14	Ideal	1	final	aerial	img	45	25231	6269	1	256	256	3
15	Ray	1	final	medical	img	7	4492	1114	1	976	976	3
16	Kraut	2	public	action	vid	4	1528	863	var	120	160	1
17	Katze	2	public	action	vid	6	1528	863	var	120	160	1
18	Kreatur	2	public	action	vid	4	1528	863	var	60	80	1
19	Ideal	2	feedback	aerial	img	45	25231	6269	1	256	256	3
20	Freddy	2	feedback	hwr	img	2	546055	136371	var	var	var	3
21	Homer	2	feedback	action	vid	12	1354	353	var	var	var	3
22	Isaac2	2	feedback	action	vid	249	38372	9561	var	102	78	1
23	Formula	2	feedback	misc.	vid	4	32994	8203	var	80	80	3
24	Apollon	2	final	people	img	100	6077	1514	1	var	var	3
25	Loukoum	2	final	hwr	img	3	27938	6939	1	var	var	3
26	Fiona	2	final	action	vid	6	8038	1962	var	var	var	3
27	Monica1	2	final	action	vid	20	10380	2565	var	168	168	3
28	Kitsune	2	final	action	vid	25	18602	4963	var	46	82	3

is fixed to $3 \times 3 \times 3$ and the pooling size and strides are both $2 \times 2 \times 2$ for max-pooling layer.

3.3. Inception-V3 with Pre-trained Weights

The Inception family uses many useful strategies to improve image classification performance. In this private baseline method, we use a pre-trained Inception V3 Szegedy et al. (2016). We resized images to 299×299. For grey images, we converted them to 3 channels and for images more than 3 channels, we extracted the RGB channels. We use pre-trained weights provided by Tensorflow and fine-tuned the model with cross entropy loss and default Adam optimizer. No data augmentation was used. Data was shuffled every epoch. The scheduling strategy is the same as before.

4. Challenge results

Both challenges (Table 1) ran for about two months each. Participation was higher (over 100 participants) in AutoCV1 than in AutoCV2 ($\simeq$ 30 participants), probably due to the higher difficulty of the second challenge and perhaps the timing. The three winners in AutoCV1 are: *kakaobrain*, *DKKimHCLee*, *base_1* and in AutoCV2: *kakaobrain*, *tanglang*, *kvr* (the full ranking can be found at `https://autodl.lri.fr/competitions/3`). More details are given on the open-sourced solutions of the winners in the next section. Progress over baseline methods is impressive (Figure 2a and Figure 2b) and beyond our expectations. Overlaid learning curves are shown in Figures 3 for a particular dataset (Caucase), illustrating that much of the contribution of the participants went into optimizing any-time learning by finding strategies to climb the learning curve fast. This is particularly dramatic on the chosen example, but is verified on other datasets. Figure 4 shows the average result over all participants for the various datasets.

5. Winning solutions

We provide here a brief overview of the methods the winners used in AutoCV1 challenge. *kakaobrain*'s solution is based on AutoAugment Cubuk et al. (2018) and they propose their own version of AutoAugment called Fast AutoAugment Lim et al. (2019) which learns the data augmentation policy in a data-driven approach. *DKKimHCLee* exploits flexible MobileNet with pretrained ImageNet weights. They aggregate different modules to deal with images of different resolutions. *base_1* team modifies the 3D CNN baseline and tries many

different choices of normalization, activation function, optimizer, regularization, etc and get finally a strong result on all datasets. In AutoCV2, the solutions of all top-3 winners (*kakaobrain*, *tanglang*, *kvr*) are based on *kakaobrain*'s code in AutoCV1. Minor modifications are made on the optimizers or the backbone networks. In the following paragraphs, we describe the methods mainly considered by top winning entries and participants' adaptations.

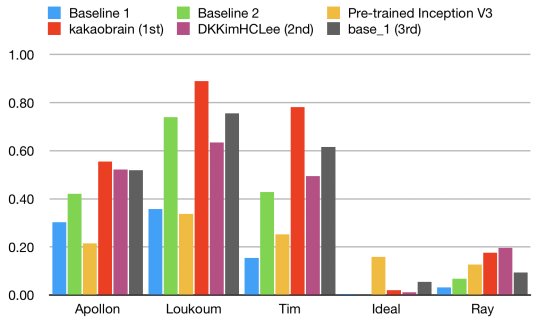
Fast AutoAugment Cubuk et al. (2018); Lim et al. (2019) Data augmentation has been shown to be an efficient regularization technique in computer vision tasks, improving generalization ability of neural networks, especially in image classification. Instead of relying on human expertise, AutoAugment formulates the search for the best augmentation policy as a discrete search problem and uses Reinforcement Learning to find the best policy. The search algorithm is implemented as a RNN controller, which samples an augmentation policy S, combining image processing operations, with their probabilities and magnitudes. S is then used to trained a child network to get a validation accuracy R, which is used to update the RNN controller by policy gradient methods. Despite a significant improvement in performance, AutoAugment requires thousands of GPU hours even with a reduced target dataset and small network.

Fast AutoAugment finds effective augmentation policies via a more efficient search strategy based on density matching between a pair of train datasets, and a policy exploration based on Bayesian optimization over stratified k-folds splits of the training dataset. The winning team *kakaobrain* implemented a light version of Fast AutoAugment, replacing the 5-folds by a single fold search and using a random search instead of Bayesian optimization (TPE) of the original paper.

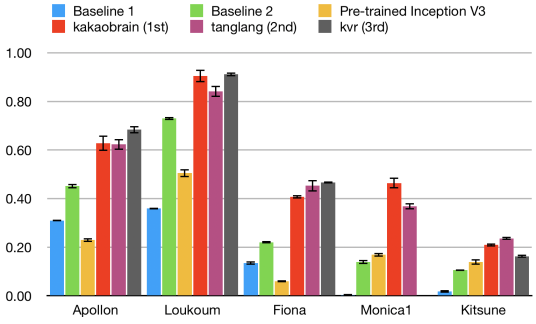
ResNet He et al. (2016)

Residual connection in CNN has been introduced in He et al. (2016) and shown to be an effective strategy to train very deep networks. ResNet consists of convolutional blocks where the input to the block is added to its output. While deep residual networks show a good performance in object recognition, shallow residual networks are still valid choices. In this regard, a number of participants have used ResNet as the backbone of their architectures in AutoCV2 challenge, specifically *kakaobrain*, *tanglang*, *DXY0808*, *ether*, *Hana.Inst.Tech* and *automl_freiburg* teams ranked as 1st, 2nd, 4th, 5th, 6th and 10th in the final phase of AutoCV2, respectively.

Most teams have used pretrained ResNet18 on Imagenet dataset. *kakaobrain* adapted input image size to the median dataset size and tuned hyperparameters offline on the public datasets. *tanglang* adapted AutoCV1 winner method to handle both 2D and 3D data while training faster. *DXY0808* adapted ResNet to handle both 2D and 3D (i.e. video) data. *ether* used AutoCV1 winner method and applied linear manifold transformation on the data. *Hana.Inst.Tech* just resized images to the network input size. *automl_freiburg* used EfficientNet Tan and Le (2019) with InceptionV2 and ResNet50 and models were tuned offline on the public datasets. All teams used L_2 norm regularization except *Hana.Inst.Tech* which used Dropout.



(a) ***AutoCV1***



(b) ***AutoCV2***

Fig. 2: **ALC scores in final phase: Winning solutions *vs.* baselines.** *AutoCV1 final phase results were not publicly revealed nor used for determining prizes, hence Apollon and Loukoum could be re-used, and no error bars were computed. Error bars in AutoCV2 are standard deviations based on 3 repeats for baselines and 9 otherwise.*

MobileNet Howard et al. (2017)

As architectures of CNN evolve, models become more and more heavy. A typical ResNet-152 will take more than 300 MB to store, thus consume much time to forward and inference. MobileNet decomposes traditional convolution operation to depthwise convolution and pointwise convolution, which largely reduce the operations required for convolution. MobileNets are based on a streamlined architecture that uses depthwise separable convolutions to build light weight deep neural networks. Howard et al. (2017) proposed two simple global hyper-parameters that efficiently trade off between latency and accuracy, allowing the model builder to choose the right sized model for their application based on the constraints of the problem. What's more, kernel numbers have been reduced by a factor. In this way, MobileNet reaches a satisfactory performance on image classfication with a much lighter model.

In the AutoCV2 challenge, teams that adapted the MobileNet as part of to their solutions are: *kvr*, *Letrain*, *team_zhaw* and *OsbornArchibald*, being ranked in 3rd, 8th, 8th and 16th position, respectivelly. Considering the *kvr* solution, they took into account picture size and dataset size while selecting network, as they observed that MobileNet works better for simple cases (such as Dataset 3 in final phase) but not on more complex ones.

5.1. Statistics on participants' approaches

After the challenges, we collected fact sheets from top participants for more details on their approach. We got 12 replies and

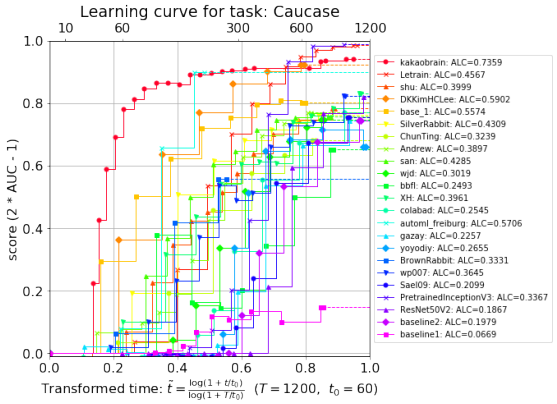


Fig. 3: Learning curves on dataset Caucase (AutoCV1 feedback phase).

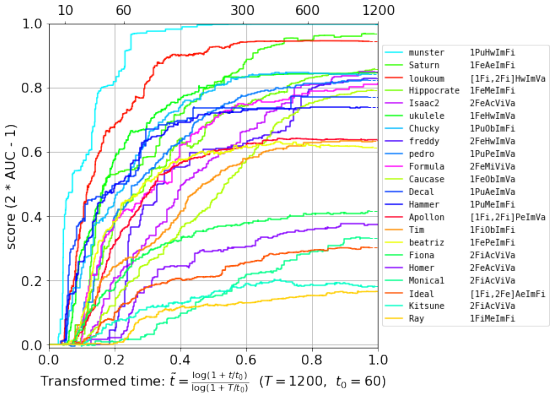


*Fig. 4: Average learning curve for each task. The legend is ordered according to final NAUC. In the legend, the label **1PuHwImFi** (for example) stands for: AutoCV**1**, **Pu**blic data, **H**and-**w**riting, **Im**age, and **Fi**xed shape. For more possible values, see Table 2.*

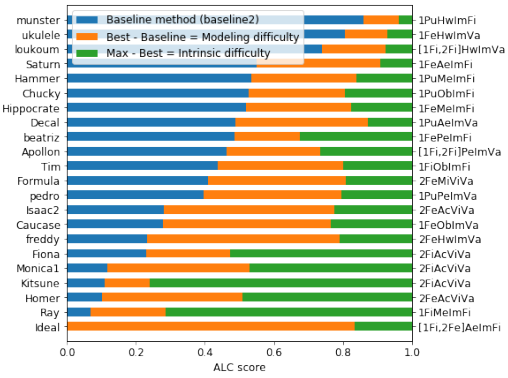


Fig. 5: Measurement of task difficulty.

present a few interesting results below. First, 100% of the participants used Deep Neural Networks to process images and videos. Although backbone networks are diverse (ResNet, MobileNet, etc), a few components are common, e.g. Batch Normalization, Dropout. Rich preprocessing methods are used (e.g. flipping, cropping, resizing, rotations) while dimensionality reductions are barely applied. 87% of the participants used ImageNet pre-trained weights in order to enable transfer learning. None of them uses Meta-Learning or XGBoost. Interestingly, although our baseline code is provided in TensorFlow, many top participants used PyTorch, which is probably due to the influence AutoCV1 winner *kakaobrain*.

The graph of Figure 6 informs on participant's effectiveness to address the *any-time learning* problem. We first factored out dataset difficulty by re-scaling ALC and NAUC scores (resulting scores on each dataset having mean 0 and variance 1). Then we plotted, for each participant, their fraction of submissions in which ALC is larger than NAUC *vs. correlation*$(ALC, NAUC)$. The size of the marker varies monotonically with average ALC. The average rank shown in the legend is computed on all datasets that their submission has been run on. The participants in the bottom half of the figure did not address well the *any-time learning* problem because their fraction of submissions in which ALC is larger than NAUC is lower than 50%. Those participants did not perform well in the challenge either (small symbols). The participants that did well in the challenge (large symbols) are all in the upper right quadrant, with both $\%(ALC > NAUC)$ larger than 50% and *correlation*$(ALC, NAUC)$ larger than 0.7. There is a small cluster at the top right corner, which includes two of the winners (kakaobraina and kvr), having both particularly high $\%(ALC > NAUC)$ and *correlation*$(ALC, NAUC)$. These participants climbed the learning curve fast and attained a good performance at the end of the time budget. Right below, there is a cluster with $\%(ALC > NAUC)$ between 0.6 and 0.8 *correlation*$(ALC, NAUC)$ between 0.7 and 0.9, corresponding to participants who were not as strong in performance at the end of the time budget, but climbed the learning curve fast.

5.2. Challenge design soundness

One concern was that $N = 5$ datasets to evaluate automated computer vision was too small to demonstrate meta-generalization In the AutoCV1 challenge, we had only one phase, thus a higher risk that the winners would overfit at the meta-generalization level: Ideally, they should deliver solutions suitable to address datasets from a wide variety of sources, not just those provided in the feed-back phase. To verify this, we prepared five additional datasets (Apollon, Tim, Ray, Ideal, Loukoum) to privately conduct a final test phase (whose results were not used towards determining the prizes). We ran the participants' submissions on these new datasets and computed the average ranking of each participant in both the feedback and the final phase. Although the ranking of participants who won the challenge was identical in both phases, the Pearson correlation coefficient between ranking vectors in both phases across all participants was only $\rho_{X,Y} = \text{cov}(X, Y)/(\sigma_X \sigma_Y) = 0.3968$ with p-value 0.0608. Some participants overfitted a lot, *i.e.* they ranked well in the feedback phase but poorly in the final phase (such as team *Letrain*, *ChunTing* and *Andrew*). These teams were mostly late entrants, meaning that they probably did not have enough time, so they might have hand-engineered many aspects of their submission. This, from another perspective, shows the goodness of the winners' submissions. We performed the same experiment for AutoCV2 in which we indeed had two phases. We obtained $\rho_{X,Y} = 0.8137$ with p-value 1.3×10^{-5}. This time we are impressed to see that for AutoCV2, there is a good correlation between feedback phase and final phase. Participants' solutions have been improved to generalize better to new data, which suggests that these solutions could be applied

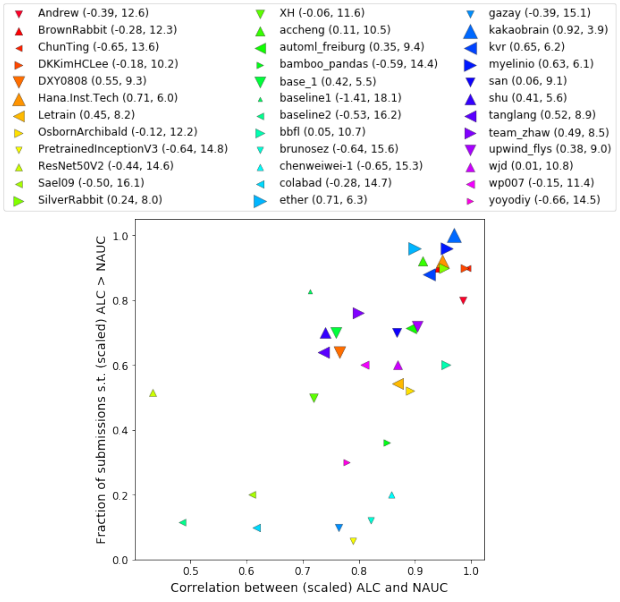


Fig. 6: ***%(ALC > NAUC)* vs *correlation(ALC, NAUC).*** *ALC and NAUC were "scaled" (see text). The numbers in the legend are average scaled ALC and average rank of each participant. The marker size increases monotonically with average scaled ALC.*

universally in all classification tasks in computer vision.

5.3. Dataset Difficulty

After the challenges terminated, another important issue for organizers is to look back to the choices of datasets. Intuitively, we want to choose datasets with low intrinsic difficulty and high modeling difficulty. There could be multiple ways of defining these two difficulties. We choose the following empirical definition: intrinsic difficulty is defined to be the maximal score (in our case, 1.0) minus the best solution's score (in our case, 1st winner's score); and the modeling difficulty is defined as the difference between the best solution's score and a baseline score (in our case, baseline 2). These scores are visualized in Figure 5. The phases in which datasets were introduced are found in Table 2. The dataset which is "ideal" for a challenge is actually called Ideal (by coincidence): it has low intrinsic difficulty and high modeling difficulty. The worst one is Kitsune (which was unfortunately in the final phase of AutoCV2). Fortunately, other final phase datasets have a modeling difficulty larger than 0.2, leaving room for the participants to improve on the baseline. Several final datasets have a large intrinsic difficulty, which may indicate that either the datasets are truly harder, or that the test conditions of the final phase are harder (a single submission with no feed-back).

6. Conclusion

The challenges AutoCV 1 and 2 have allowed us to demonstrate that fully automated computer vision is not as far away as many thought it was. Indeed, when we started laying the basis of the challenge design nearly three years ago, our collaborators at Google believed that this challenge setting was way too hard to deliver results. Three years later, we are pleased to see that the research community was able to deliver on this problem, in spite of its great difficulty. Remarkably, we have now publicly available software capable of handling any image or video classification task without any human intervention whatsoever. In order to provide with a large but treatable amount of data, we invested a significant effort in data preparation and formatting. Spatial and temporal sampling as well as image compression were applied in order to define a fixed maximum size for challenge datasets. Although future work includes the analysis of larger datasets, with the current size participants did a great job providing good autonomous solutions with limited hardware resources and assigned time budget. How "generalisable" the solutions are to significantly larger datasets remains to be determined. However, the overall methodology developed should be applicable in a broader range of data size and difficulty: (1) Re-use a pre-trained deep neural network whose architecture has a proven track record in computer vision; (2) Use of data "augmentation" (or distillation in the first part of the learning curve); (3) Use of modularity in training strategies to sparingly allocate training time where and when needed to climb the learning curve fast and smoothly. As organizers, we learned some lessons that will facilitate the organization of future challenges in the series, including preventing duplicate accounts to avoid some participants to abuse computing resources, and designing better fact sheets to collect more informative data on winning entries. Our AutoDL challenge series includes several other on-going or planned challenges; AutoNLP, AutoSpeech, and AutoWSL (weakly supervised learning), culminating in the final AutoDL challenge planned for the end of 2019 (officially part of the NeurIPS 2019 challenge series).

References

Cubuk, E.D., Zoph, B., Mane, D., Vasudevan, V., Le, Q.V., 2018. Autoaugment: Learning augmentation policies from data. *τarXiv* : 1805.09501.

He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition, in: CVPR, IEEE. pp. 770–778.

Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H., 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *τarXiv* : 1704.04861.

Kingma, D.P., Ba, J., 2014. Adam: A Method for Stochastic Optimization *τarXiv* : 1412.6980.

Lim, S., Kim, I., Kim, T., Kim, C., Kim, S., 2019. Fast autoaugment. *τarXiv* : 1905.00397.

Liu, Z., Guyon, I., Jacques Jr., J., Madadi, M., Escalera, S., Pavao, A., Escalante, H.J., Tu, W.W., Xu, Z., Treguer, S., 2019. AutoCV Challenge Design and Baseline Results, in: CAp 2019, Toulouse, France. URL: `https://hal.archives-ouvertes.fr/hal-02265053`.

Pham, H., Guan, M.Y., Zoph, B., Le, Q.V., Dean, J., 2018. Efficient Neural Architecture Search via Parameter Sharing *τarXiv* : 1802.03268.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z., 2016. Rethinking the Inception Architecture for Computer Vision, in: CVPR, IEEE, Las Vegas, NV, USA. pp. 2818–2826. doi:`10.1109/CVPR.2016.308`.

Tan, M., Le, Q., 2019. Efficientnet: Rethinking model scaling for convolutional neural networks, in: ICML, pp. 6105–6114.

Zoph, B., Le, Q.V., 2016. Neural architecture search with reinforcement learning *τarXiv* : 1611.01578.